# REGULARIZATION

SimpleAI.ch

## L1 REGULARIZATION (LASSO)

**least absolute shrinkage and selection operator**

$$loss = error(y, \hat{y}) + \lambda \sum_j |\beta_j|$$

- Prevents the weights from getting too large (defined by L1 norm).
- L1 reg. introduces sparsity in the weights. It forces more weights to be zero

LASSO do well if there are few significant predictors and the magnitudes of the others are close to zero

**LASSO LIMITATIONS**

- **If p>>n**, where p is large, and n "small", == LASSO selects at most n variables before it saturates
- **No group selection:** if there is a group of highly correlated variables == LASSO selects only one such var. and ignores the rest

## L2 REGULARIZATION (RIDGE)

$$loss = error(y, \hat{y}) + \lambda \sum_j \beta_j^2$$

- **Prevents the weights from getting too large (defined by L2 norm).**
- Developed as the solution to the imprecision of least square estimators when linear regression models have some multicollinearity (highly correlated features).

**L2 LIMITATIONS**

- includes all the predictors in the final model (also a pros)
- == unable to perform feature selection.
- Shrinks coeff^s towards zero.

Ridge regression works well if there are many predictors of about the same magnitude. This means all predictors have similar power to predict the target value.

## ELASTIC NET

**includes the LASSO and ridge regression in penalty term with lambda1 &2, or their ratio**

- unique minimum (strictly convex) - The quadratic penalty term makes the loss function strongly convex
- Implemented. In skleanr for linear regression, logistic regression and linear support vector machines

$$argmin_\beta \sum_i (y_i - \boldsymbol{\beta}'\boldsymbol{x}_i)^2 + \lambda_1 \sum_{k=1}^{K} |\beta_k| + \lambda_2 \sum_{k=1}^{K} \beta_k^2$$

The $\lambda_1$ is Lasso penalty (L1) and $\lambda_2$ is the Ridge regression penalty (L2)

## LARS

**least-angle regression (LARS)**

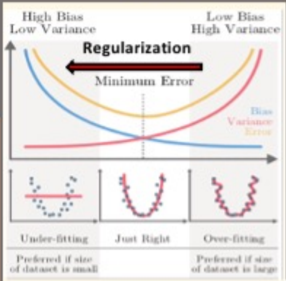
- It is an algorithm for fitting linear regression models to high-dimensional data
- Allows selection of variables and their coefficients to include in the model

- **Effective for p>>n datasets**
- If 2 var.s are equally correlated with the response var., it will provides similar weights for both of them

**LARS LIMITATIONS**

- Sensitive to collinearity

## HOW REGULARIZATION WORKS?



**It combats over-fitting by reducing model variance and complexity**

- **The main idea,** is to add constraint to the amplitude od the coefficients to cost function, called **PENALTY, or penalization term**. It is a Procedure that allows, to keep weights relatively small, or select most relevant coefficients.
- to control the regularization strength, we are using **ALFA/LAMBDA** hyperparameter [0,1],

**Larger the weights == more complex model == more chances of overfitting**

### ALFA/LAMBDA

- **Controls the strength of the penalty term**
- controls the amount of attention that the learning process should pay to the penalty == the amount to penalize the model based on the size of the weights.
- value between 0.0 (no penalty) and 1.0 (full penalty)
  - **0.0 == low bias (high variance),**
  - **1.0 == high bias (low variance).**

If the penalty is too strong, the model will underestimate the weights and underfit the problem. If the penalty is too weak, the model will be allowed to overfit the training data.

### FEATURE SELECTION

- **LassoCV** is most often preferable for high-dimensional datasets with many collinear features
- **LassoLarsCV** allows exploring more relevant values of alpha parameter then Lasso method, and, if the n<<p, it is often faster than LassoCV.

### SHOULD I EVEN USE LASSO ?

**Comments that I found on stack overflow:** Even in the case when you have a strong reason to use L1, given the number of features, I would recommend going for Elastic Nets instead. Granted this will only be a practical option if you are doing linear/logistic regression. But, in that case, Elastic Nets have proved to be (in theory and in practice) better than L1/Lasso. Elastic Nets combine L1 and L2 regularization at the "only" cost of introducing another hyperparameter to tune (see Hastie's paper for more details Page on stanford.edu). Even in a situation where you might benefit from L1's sparsity in order to do feature selection, using L2 on the remaining variables is likely to give better results than L1 by itself.

### WHEN TO USE EACH TYPE

- **Lasso** — data have few significant predictors and the magnitudes of the others are close to zero
- **Ridge** - there are many large predictors of about the same value
  - multicolinearity
- **ElasticNet** —there are multiple features which are correlated with one another. Lasso is likely to pick one of these at random, while elastic-net is likely to pick both.
- **LARS** - alternative to ridge and ElasticNet for multiple colinear features,
  - iterative approach,
  - More effective for p>>n datasets, and alfa exploration

## What else beside regularization can be used to improve your model?

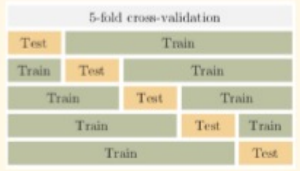
### APPLY PROPER DASTA TRANSFORMATIONS

- Scaling/normalization/
- Binarization/Kbins selection
- Non-linear transformations
- Add polynomial features,

### MODIFY SAMPLING STRATEGY

- Data Augmentation
- Synthetic Data
- K-fold Cross-Validation (CV)

Divide the data into k groups. Train on (k-1) groups and test on 1 group. Try all k possible combinations.



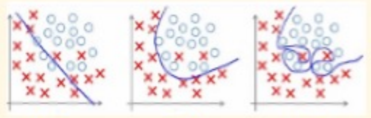
### USE MODEL PARAMETERS TO REDUCE. VARIANCE

**E1. k-nearest neighbours (k-nn)**

Generally k-nn alg has low bias & high variance, but the trade-off can be changed **by increasing the k value** which increases the number of neighbours that contribute to the prediction and in turn increases the bias of the model.

**E 2. The support vector machine (SVM)**

has low bias and high variance, but the trade-off can be changed by **increasing the C parameter** that influences the number of violations of the margin allowed in the training data which increases the bias but decreases the variance.



k=5 k=2 k=0

### MODIFY TRAINING APPROACH

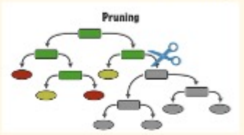
**Noise Injection**

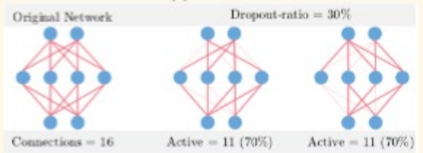
- Add random noise to the weights when they are being learned.
- It pushes the model to be relatively insensitive to small variations in the weights, hence regularization

**Dropout**

- Generally used for neural networks.
- Connections between consecutive layers are randomly dropped based on a dropout-ratio and the remaining network is trained in the current iteration. In the next iteration, another set of random connections are dropped.

**Pruning**




Original Network
Dropout-ratio = 30%
Connections = 16
Active = 11 (70%)
Active = 11 (70%)